

09/637,532

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant: Dmitriy Rumynin et al.

Examiner: David H. Malzahn

Serial No.: 09/637,532

Group Art Unit: 2124

Filed: August 11, 2000

Docket: 1365.033US1

Title: A PARALLEL COUNTER AND MULTIPLICATION LOGIC CIRCUIT

**COMMUNICATION REGARDING FILING OF PRIORITY
DOCUMENT IN ACCORDANCE WITH 35 U.S.C. 119**

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

RECEIVED

OCT 21 2003

Dear Sir:

Technology Center 2100

In accordance with the requirements for claiming right of priority under 35 U.S.C. 119, enclosed for filing in connection with the above-identified application is a certified copy of Applicant's prior application, Great Britain Application No. 0019287.2, filed on August 4, 2000.

Respectfully submitted,

DMITRIY RUMYNIN ET AL.

By their Representatives,

SCHWEGMAN, LUNDBERG, WOESSNER & KLUTH, P.A.
P.O. Box 2938
Minneapolis, MN 55402
(612) 349-9592

Date Oct. 1, 2003 By Ann M. McCrackin
Ann M. McCrackin
Reg. No. 42,858

CERTIFICATE UNDER 37 CFR 1.8: The undersigned hereby certifies that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail, in an envelope addressed to: Commissioner of Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on this 1st day of October, 2003.

PATRICIA A. HULTMAN
Name

Signature

THIS PAGE BLANK (USPTO)



INVESTOR IN PEOPLE

The Patent Office
Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation and Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein together with the Statement of inventorship and of right to grant of a Patent (Form 7/77), which was subsequently filed.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, C. or PLC.

Registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.

**CERTIFIED COPY OF
PRIORITY DOCUMENT**

Signed *AmBrewer*.

Dated 18 October 2001

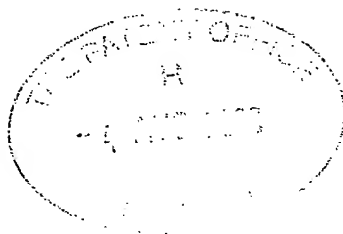
THIS PAGE BLANK ~~WIPED~~

Request for grant of a patent

(See the notes on the back of this form. You can also get an explanatory leaflet from the Patent Office to help you fill in this form)

The Patent Office

Cardiff Road
Newport
Gwent NP9 1RH



1. Your reference **GBP82563**

2. Patent application number
(The Patent Office will fill in this part)

3 Full name, address and postcode of the or of each applicant (underline all surnames)

Automatic Parallel Designs Limited
39 Lyonsdown Road
New Barnet
Hertfordshire EN5 1LE

0019287.2

7422173cc

Patents ADP number (if you know it)

If the applicant is a corporate body, give the country/state of its incorporation

United Kingdom

4. Title of the invention **Parallel counter**

5. Name of your agent (if you have one)
"Address for service" in the United Kingdom to which all correspondence should be sent (including the postcode)

Marks & Clerk
57 - 60 Lincolns Inn Fields
London WC2A 3LS

Patents ADP number (if you know it)

18001

6.If you are declaring priority from one or more earlier patent applications, give the country and the date of filing of the or of each of these earlier applications and (if you know it) the or each application number

Country

Priority application No
(if you know it)

Date of filing
(day / month / year)

7.If this application is divided or otherwise derived from an earlier UK application, give the number and the filing date of the earlier application

Number of earlier application

Date of filing
(day / month / year)

8. Is a statement of inventorship and of right to grant of a patent required in support of this request? (Answer 'Yes' if:
a) any applicant named in part 3 is not an inventor, or
b) there is an inventor who is not named as an applicant, or
c) any named applicant is a corporate body.
See note (d))

Yes

Patents Form 1/77

9. Enter the number of sheets for any of the following items you are filing with this form. Do not count copies of the same document

Continuation sheets of this form	0
Description	16
Claim(s)	4 871
Abstract	1
Drawing(s)	18 + 18

10. If you are also filing any of the following, state how many against each item.

Priority documents

Translations of priority documents

Statement of inventorship and right to grant of a patent (Patents Form 7/77) 1

Request for preliminary examination and search (Patents Form 9/77) 1

Request for substantive examination (Patents Form 10/77)

Any other documents (please specify)

11. I/We request the grant of a patent on the basis of this application.

Signature

Markus D. Acker

Date: 4 August 2000

12. Name and daytime telephone number of person to contact in the United Kingdom

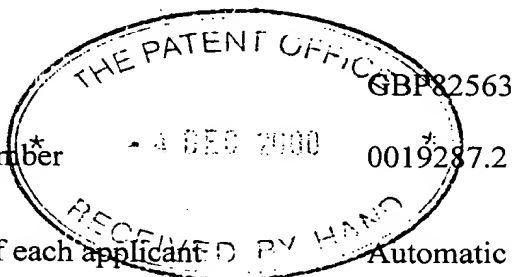
GB Patent Filings
0207 400 3000

**Statement of inventorship and of
right to grant of a patent**

The Patent Office

Cardiff Road
Newport
South Wales
NP9 1RH

1. Your reference



2. Patent application number
(if you know it)

0019287.2

3. Full name of the or of each applicant

Automatic Parallel Designs Limited

4. Title of the invention

Parallel Counter

5. State how the applicant(s) derived the right
from the inventor(s) to be granted a patent

By virtue of employment

6. How many, if any, additional Patents Forms
7/77 are attached to this form?
(see note (c))

7.

I/We believe that the person(s) named over the page
(and on any extra copies of this form) is/are the inventor(s) of the
invention which the above patent application relates to.

Signature

Date

Marks & Clerk

1/12/00

8. Name and daytime telephone number of
person to contact in the United Kingdom

Marks & Clerk

0207 400 3000

Notes

- a) If you need help to fill in this form or you have any questions, please contact the Patent Office on 0645 500505.
- b) Write your answers in capital letters using black ink or you may type them.
- c) If there are more than three inventors, please write the names and addresses of the other inventors on the back of another Patents Form 7/77 and attach it to this form.
- d) When an application does not declare any priority, or declares priority from an earlier UK application, you must provide enough copies of this form so that the Patent Office can send one to each inventor who is not an applicant.
- e) Once you have filled in the form you must remember to sign and date it.

Patents Form 7/77

Enter the full names, addresses and postcodes of the inventors in the boxes and underline the surnames

Rumynin, Dmitriy
258 Sir Henry Parkes Road
Coventry CV4 8GG

7955040001

Talwar, Sunil
26a Binswood Avenue
Leamington Spa
Warwickshire CV32 5SQ

CSC 36097001

Meulemans, Peter
1 Latham Road
Coventry CV5 6HR

08036105001

Patents ADP number (if you know it):

Reminder
Have you signed the form?

4/08/00

A PARALLEL COUNTER AND A MULTIPLICATION LOGIC CIRCUIT

The present invention generally relates to digital electronic devices and more particularly to a digital electronic device performing binary logic. In one aspect the present invention relates to a parallel counter and in another aspect the present invention relates to a multiplication logic circuit for multiplying two binary numbers.

It is instrumental for many applications to have a block that adds n inputs together. An output of this block is a binary representation of the number of high inputs. Such blocks, called parallel counters (L. Dadda, *Some Schemes for Parallel Multipliers*, Alta Freq **34**: 349-356 (1965); E. E. Swartzlander Jr., *Parallel Counters*, IEEE Trans. Comput. **C-22**: 1021-1024 (1973)), are used in circuits performing binary multiplication. There are other applications of a parallel counter, for instance, majority-voting decoders or RSA encoders and decoders. It is important to have an implementation of a parallel counter that achieves a maximal speed. It is known to use parallel counters in multiplication (L. Dadda, *On Parallel Digital Multipliers*, Alta Freq **45**: 574-580 (1976)).

A full adder is a special parallel counter with a three-bit input and a two-bit output. A current implementation of higher parallel counters i.e. with a bigger number of inputs is based on using full adders (C. C. Foster and F. D. Stockton, *Counting Responders in an Associative Memory*, IEEE Trans. Comput. **C-20**: 1580-1583 (1971)). In general, the least significant bit of an output is the fastest bit to produce in such implementation while other bits are usually slower.

The following notation is used for logical operations:

\oplus - Exclusive OR;

\vee - OR;

\wedge - AND;

\neg - NOT.

An efficient prior art design (Foster and Stockton) of a parallel counter uses full adders. A full adder, denoted FA, is a three-bit input parallel counter shown in figure 1. It has three inputs X_1, X_2, X_3 , and two outputs S and C . Logical expressions for outputs are

$$S = X_1 \oplus X_2 \oplus X_3,$$

$$C = (X_1 \wedge X_2) \vee (X_1 \wedge X_3) \vee (X_2 \wedge X_3).$$

A half adder, denoted HA, is a two bit input parallel counter shown in figure 1. It has two inputs X_1, X_2 and two outputs S and C . Logical expressions for outputs are

$$S = X_1 \oplus X_2,$$

$$C = X_1 \wedge X_2.$$

A prior art implementation of a seven-bit input parallel counter illustrated in figure 2.

Multiplication is a fundamental operation. Given two n -digit binary numbers

$$A_{n-1}2^{n-1} + A_{n-2}2^{n-2} + \dots + A_12 + A_0 \text{ and } B_{n-1}2^{n-1} + B_{n-2}2^{n-2} + \dots + B_12 + B_0,$$

their product

$$P_{2n-1}2^{2n-1} + P_{2n-2}2^{2n-2} + \dots + P_12 + P_0$$

may have up to $2n$ digits. Logical circuits generating all P_i as outputs generally follow the scheme in figure 14. Wallace has invented the first fast architecture for a multiplier, now called the Wallace-tree multiplier (Wallace, C. S., *A Suggestion for a Fast Multiplier*, IEEE Trans. Electron. Comput. EC-13: 14-17 (1964)). Dadda has investigated bit behaviour in a multiplier (L. Dadda, *Some Schemes for Parallel Multipliers*, Alta Freq 34: 349-356 (1965)). He has constructed a variety of multipliers and most multipliers follow Dadda's scheme.

Dadda's multiplier uses the scheme in on figure 14. If inputs have 8 bits then 64 parallel AND gates generate an array shown in figure 15. The AND gate sign \wedge is omitted for clarity so that $A_i \wedge B_j$ becomes $A_i B_j$. The rest of figure 15 illustrates array reduction that involves full adders (FA) and half adders (HA). Bits from the same column are added by half adders or full adders. Some groups of bits fed into a full adder are in rectangles. Some groups of bits fed into a half adder are in ovals. The result of array reduction is

just two binary numbers to be added at the last step. One adds these two numbers by one of fast addition schemes, for instance, conditional adder or carry-look-ahead adder.

In accordance with the first aspect the present invention provides a parallel counter which is based on algebraic properties of symmetric functions. Each of the plurality of binary output bits is generated as a symmetric function of a plurality of binary input bits.

The symmetric functions comprise logically AND combining sets of one or more binary inputs and logically OR or exclusive OR logic combining the logically combined sets of binary inputs to generate a binary output. The OR and the exclusive OR symmetric functions are elementary symmetric functions and the generated output binary bit depends only on the number of high inputs among the input binary bits. For the OR symmetric function, if the number of high inputs is m , the output is high if and only if $m \geq k$, where k is the size of the sets of binary inputs. Similarly, the generated output binary bit using the exclusive OR symmetric function is high if and only if $m \geq k$ and the number of subsets of inputs of the set of high inputs is an odd number. The size of the sets can be selected. The i^{th} output bit can be generated using the symmetric function using exclusive OR logic by selecting the set sizes to be of size 2^i , where i is an integer from 1 to N , N is the number of binary outputs, and i represents the significance of each binary output.

The sets of binary inputs used in the symmetric functions are each unique and they cover all possible combinations of binary inputs.

Thus in one embodiment of the present invention, each of the binary outputs can be generated using a symmetric function which uses exclusive OR logic. However, exclusive OR logic is not as fast as OR logic.

Thus in accordance with an embodiment of the present invention at least one of the binary outputs is generated as a symmetric function of the binary inputs using OR logic for combining a variety of sets of one or more binary inputs. The logic is arranged to

logically AND members of each set of binary inputs and logically OR the result of the AND operations.

Thus use of the symmetric function using OR logic is faster and can be used for generation of the most significant output bit. In such an embodiment the set size is set to be 2^{N-1} , where N is the number of binary outputs and the N^{th} binary output is the most significant.

It is also possible to use the symmetric function using OR logic for less significant bits on the basis of the output value of a more significant bit. In such a case, a plurality of possible binary outputs for a binary output less significant than the N^{th} are generated as symmetric functions of the binary inputs using OR logic for combining a plurality of sets of one or more binary inputs, where N is the number of binary outputs. Selector logic is provided to select one of the possible binary outputs based on a more significant binary output value. The size of the sets used in such an arrangement for the $(N-1)^{\text{th}}$ bit is preferably $2^{N-1} + 2^{N-2}$ and 2^{N-2} respectively and one of the possible binary outputs is selected based on the N^{th} binary output value.

In one embodiment of the present invention the circuit is designed in a modular form. A plurality of subcircuit logic modules are designed, each for generating intermediate binary outputs as a symmetric function of some of the binary inputs. Logic is also provided in this embodiment for logically combining the intermediate binary outputs to generate a binary outputs.

Since OR logic is faster, in a preferred embodiment the subcircuit logic modules implement the symmetric functions using OR logic. In one embodiment the subcircuit modules can be used for generating some binary outputs and one or more logic modules can be provided for generating other binary outputs in which each logic module generates a binary output as a symmetric function of the binary inputs exclusive OR logic for combining a plurality of sets of one or more binary inputs.

Thus this aspect of the present invention provides a fast circuit that can be used in any architecture using parallel counters. The design is applicable to any type of technology from which the logic circuit is built.

The parallel counter in accordance with this aspect of the present invention is generally applicable and can be used in a multiplication circuit that is significantly faster than prior art implementations.

In accordance with the second aspect of the present invention a technique for multiplying $2N$ bit binary numbers comprises an array generation step in which an array of logical combinations between the bits of the two binary numbers is generated which is of reduced size compared to the prior art.

In accordance with this aspect of the present invention, a logic circuit for multiplying $2N$ bit numbers comprises array generation logic for performing the logical AND operation between each bit in one binary bit and each bit in the other binary number to generate an array of logical AND combinations comprising an array of binary values, and for further logically combining logically adjacent values to reduce the maximum depth of the array to below N bits; array reduction logic for reducing the depth of the array to two binary numbers; and addition logic for adding the binary values of the two binary numbers.

When two binary numbers are multiplied together, as is conventional, each bit A_i of the first binary number is logically AND combined with each bit B_j of the second number to generate the array which comprises a sequence of binary numbers represented by the logical AND combinations, A_i AND B_j . The further logical combinations are carried out by logically combining the combinations A_1 AND B_{N-2} , A_1 AND B_{N-1} , A_0 AND B_{N-2} , and A_0 AND B_{N-1} , where N is the number of bits in the binary numbers. In this way the size of the maximal column of numbers to be added together in the array is reduced.

More specifically the array generation logic is arranged to combine the combinations A_1 AND B_{n-2} and A_0 AND B_{n-1} using exclusive OR logic to replace these combinations and to combine A_1 AND B_{n-1} and A_0 AND B_{n-2} to replace the A_1 AND B_{n-1} combination.

In one embodiment of the present invention the array reduction logic can include at least one of: at least one full adder, at least one half adder, and at least one parallel counter. The or each parallel counter can comprise the parallel counter in accordance with the first aspect of the present invention.

The second aspect of the present invention provides a reduction of the maximal column length in the array thereby reducing the number of steps required for array reduction. When the first aspect of the present invention is used in conjunction with the second aspect of the present invention, an even more efficient multiplication circuit is provided.

Embodiments of the present invention will now be described with reference to the accompanying drawings, in which:

Figure 1 is a schematic diagram of a full adder and a half adder in accordance with the prior art,

Figure 2 is a schematic diagram of a parallel counter using full adders in accordance with the prior art,

Figure 3 is a schematic diagram illustrating the logic modules executing the symmetric functions for the generation of binary outputs and the multiplexor (selector) used for selecting outputs,

Figure 4 is a diagram illustrating the logic for implementing the symmetric function OR_3_1 ,

Figure 5 is a diagram illustrating the logic for implementing the symmetric function OR_4_1 ,

Figure 6 is a diagram illustrating the logic for implementing the symmetric function OR_5_1 using 2 3 input OR gates,

Figure 7 is a diagram illustrating the logic for implementing the symmetric function $EXOR_7_1$ using two input exclusive OR gates,

Figure 8 is a diagram illustrating the logic for implementing the symmetric function OR_3_2,

Figure 9 is a diagram illustrating the logic for implementing the symmetric function EXOR_5_3,

Figure 10 is a diagram illustrating a parallel counter using the two types of symmetric functions and having seven inputs and three outputs,

Figure 11 is a diagram illustrating splitting of the symmetric function OR_7_2 into sub modules to allow the reusing of smaller logic blocks,

Figure 12 is a diagram of a parallel counter using the EXOR_7_1 symmetric function for the generation of the least significant output bit from all of the input bits, and smaller modules implementing symmetric functions using OR logic to generate the second and third output bits,

Figure 13 is a another diagram of a parallel counter similar to that of Figure 12 accept that the partitioning of the inputs is chosen differently to use different functional sub modules,

Figure 14 is a diagram of the steps used in the prior art for multiplication,

Figure 15 is a schematic diagram of the process of Figure 14 in more detail,

Figure 16 is a diagram illustrating the properties of diagonal regions in the array,

Figure 17 is a diagram illustrating array deformation in accordance with the embodiment of the present invention and the subsequent steps of array reduction and adding, and

Figure 18 is a diagram of logic used in this embodiment for array generation.

The first aspect of the present invention will now be described.

The first aspect of the present invention relates to a parallel counter counting the number of high values in a binary number. The counter has i outputs and n inputs where i is determined as being the integer part of $\log_2 n$ plus 1

A mathematical basis for the first aspect of the present invention is a theory of symmetric functions. We denote by C_k^n the number of distinct k element subsets of a set

of n elements. We consider two functions $EXOR_n_k$ and OR_n_k of n variables X_1, X_2, \dots, X_n given by

$$EXOR_n_k(X_1, X_2, \dots, X_n) = \bigoplus (X_{i1} \wedge X_{i2} \wedge \dots \wedge X_{ik}),$$

$$OR_n_k(X_1, X_2, \dots, X_n) = \bigvee (X_{i1} \wedge X_{i2} \wedge \dots \wedge X_{ik})$$

where $(i1, i2, \dots, ik)$ runs over all possible subsets of $\{X_1, X_2, \dots, X_n\}$ that contain precisely k elements. Blocks that produce such outputs are shown on figure 3.

The functions $EXOR_n_k$ and OR_n_k are elementary symmetric functions. Their values depend only on the number of high inputs among $X_1, X_2, X_3, \dots, X_n$. More precisely, if m is the number of high inputs among $X_1, X_2, X_3, \dots, X_n$ then $OR_n_k(X_1, X_2, \dots, X_n)$ is high if and only if $m \geq k$. Similarly, $EXOR_n_k(X_1, X_2, \dots, X_n)$ is high if and only if $m \geq k$ and C_k^m is odd.

Although $EXOR_n_k$ and OR_n_k look similar, OR_n_k is much faster to produce since EXOR-gates are slower than OR-gates.

In the above representation n is the number of inputs and k is the size of the subset of inputs selected. Each set of k inputs is a unique set and the subsets comprise all possible subsets of the set of inputs. For example, the symmetric function OR_3_1 has three inputs X_1, X_2 and X_3 and the set size is 1. Thus the sets comprise X_1, X_2 and X_3 . Each of these sets is then logically OR combined to generated the binary output. The logic for performing this function is illustrated in Figure 4.

Figure 5 illustrates the logic for performing the symmetric OR_4_1 .

When the number of inputs become large, it may not be possible to use simple logic.

Figure 6 illustrates the use of two OR gates for implementing the symmetric function OR_5_1 .

Figure 7 similarly illustrates the logic for performing EXOR_7_1. The sets comprise the inputs $X_1, X_2, X_3, X_4, X_5, X_6$, and X_7 . These inputs are input into three levels of exclusive OR gates.

When k is greater than 1, the inputs in a subset must be logically AND combined. Figure 8 illustrates logic for performing the symmetric function OR_3_2. The inputs X_1 and X_2 comprise the first set and are input to a first AND gate. The inputs X_1 and X_3 constitute a second set and are input to a second AND gate. The inputs X_2 and X_3 constitute a third set and are input to a third AND gate. The output of the AND gates are input to an OR gate to generate the output function.

Figure 9 is a diagram illustrating the logic for performing the symmetric function EXOR_5_3. To perform this function the subsets of size 3 for the set of five inputs comprise ten sets and ten AND gates are required. The output of the AND gates are input to an exclusive OR gate to generate the function.

The specific logic to implement the symmetric functions will be technology dependent. Thus the logic can be designed in accordance with the technology to be used.

In accordance with a first embodiment of the present invention the parallel counter of each output is generated using a symmetric function using exclusive OR logic.

Let the parallel counter have n inputs X_1, \dots, X_n and $t+1$ outputs S_t, S_{t-1}, \dots, S_0 . S_0 is the least significant bit and S_t is the most significant bit. For all i from 0 to t ,

$$S_i = \text{EXOR_n_2}^i(X_1, X_2, \dots, X_n).$$

It can thus be seen that for a seven bit input i.e. $n=7$, i will have values of 0, 1 and 2. Thus to generate the output S_0 the function will be EXOR_7_1, to generate the output S_1 the function will be EXOR_7_2 and to generate the output S_2 the function will be EXOR_7_4. Thus for the least significant bit the set size (k) is 1, for the second bit the

set size is 2 and for the most significant bit the set size is 4. Clearly the logic required for the more significant bits becomes more complex and thus slower to implement.

Thus in accordance with a second embodiment of the present invention, the most significant output bit is generated using a symmetric function using OR logic.

This is more practical since OR_{n_k} functions are faster than EXOR_{n_k} functions. For the most significant output bit

$$S_k = \text{OR}_{n_2^t}(X_1, X_2, \dots, X_n).$$

In particular, with a seven-bit input

$$S_2 = \text{OR}_{7_4}(X_1, X_2, X_3, X_4, X_5, X_6, X_7).$$

Thus in this second embodiment of the present invention the most significant bit is generated using symmetric functions using OR logic whereas the other bits are generated using symmetric functions which use exclusive OR logic.

A third embodiment will now be described in which intermediate bits are generated using symmetric functions using OR logic.

An arbitrary output bit can be expressed using OR_{n_k} functions if one knows bits that are more significant. For instance, the second most significant bit is given by

$$S_{t-1} = (S_t \wedge \text{OR}_{n_2^{t-1}}) \vee ((\neg S_t) \wedge \text{OR}_{n_2^{t-1}}).$$

In particular, with a seven-bit input

$$S_1 = (S_2 \wedge \text{OR}_{7_6}(X_1, X_2, X_3, X_4, X_5, X_6, X_7)) \vee ((\neg S_2) \wedge \text{OR}_{7_2}(X_1, X_2, X_3, X_4, X_5, X_6, X_7)).$$

A further reduction is

$$S_1 = \text{OR}_{7_6}(X_1, X_2, X_3, X_4, X_5, X_6, X_7) \vee ((\neg S_2) \wedge \text{OR}_{7_2}(X_1, X_2, X_3, X_4, X_5, X_6, X_7)).$$

A multiplexer MU, shown in figure 3, implements this logic. It has two inputs X_0, X_1 , a control C , and an output Z determined by the formula

$$Z = (C \wedge X_1) \vee ((\neg C) \wedge X_0).$$

It is not practical to use either EXOR_{n_k} functions or OR_{n_k} functions exclusively. It is optimal to use OR_{n_k} functions for a few most significant bits and EXOR_{n_k} functions for the remaining bits. The fastest, in TSMC.25, parallel counter with 7 inputs is shown in figure 10.

Future technologies that have fast OR_{15_8} blocks would allow building a parallel counter with 15 inputs. A formula for the third significant bit using OR_{n_m} functions is thus:

$$S_{t-2} = (S_t \wedge S_{t-1} \wedge \text{OR}_{n_2^t+2^{t-1}+2^{t-2}}) \vee (S_t \wedge (\neg S_{t-1}) \wedge \text{OR}_{n_2^t+2^{t-2}}) \vee ((\neg S_t) \wedge S_{t-1} \wedge \text{OR}_{n_2^{t-1}+2^{t-2}}) \vee ((\neg S_t) \wedge (\neg S_{t-1}) \wedge \text{OR}_{n_2^{t-2}}).$$

A fourth embodiment of the present invention will now be described which divides the logic block implementing the symmetric function into small blocks which can be reused.

An implementation of OR_{7_2} is shown in figure 11. The 7 inputs are split into two groups: five inputs from X_1 to X_5 and two remaining inputs X_6 and X_7 . Then the following identity is a basis for the implementation in figure 11.

$$\begin{aligned} \text{OR}_{7_2}(X_1, \dots, X_7) &= \text{OR}_{5_2}(X_1, \dots, X_5) \vee \\ &(\text{OR}_{5_1}(X_1, \dots, X_5) \wedge \text{OR}_{2_1}(X_6, X_7)) \vee \text{OR}_{2_2}(X_6, X_7) \end{aligned}$$

One can write similar formulas for OR_{7_4} and OR_{7_6}. Indeed,

$$\begin{aligned} \text{OR}_{7_4}(X_1, \dots, X_7) &= \text{OR}_{5_4}(X_1, \dots, X_5) \vee \\ &(\text{OR}_{5_3}(X_1, \dots, X_5) \wedge \text{OR}_{2_1}(X_6, X_7)) \vee \\ &(\text{OR}_{5_2}(X_1, \dots, X_5) \wedge \text{OR}_{2_2}(X_6, X_7)), \\ \text{OR}_{7_6}(X_1, \dots, X_7) &= \\ &(\text{OR}_{5_5}(X_1, \dots, X_5) \wedge \text{OR}_{2_1}(X_6, X_7)) \vee \\ &(\text{OR}_{5_4}(X_1, \dots, X_5) \wedge \text{OR}_{2_2}(X_6, X_7)). \end{aligned}$$

Thus, it is advantageous to split variables and reuse smaller OR_{n_k} functions in a parallel counter. For instance, an implementation of a parallel counter based on partitioning seven inputs into groups of two and five is in figure 12.

Similarly, one can partition seven inputs into groups of four and three. An implementation of the parallel counter based on this partition is in figure 13. One uses the following logic formulas in this implementation.

$$\begin{aligned}
 \text{OR_7_2}(X_1, \dots, X_7) &= \text{OR_4_2}(X_1, X_2, X_3, X_4) \vee \\
 &(\text{OR_4_1}(X_1, X_2, X_3, X_4) \wedge \text{OR_3_1}(X_5, X_6, X_7)) \vee \text{OR_3_2}(X_5, X_6, X_7), \\
 \text{OR_7_4}(X_1, \dots, X_7) &= \text{OR_4_4}(X_1, X_2, X_3, X_4) \vee \\
 &(\text{OR_4_3}(X_1, X_2, X_3, X_4) \wedge \text{OR_3_1}(X_5, X_6, X_7)) \vee \\
 &(\text{OR_4_2}(X_1, X_2, X_3, X_4) \wedge \text{OR_3_2}(X_5, X_6, X_7)) \vee \\
 &(\text{OR_4_1}(X_1, X_2, X_3, X_4) \wedge \text{OR_3_3}(X_5, X_6, X_7)), \\
 \text{OR_7_6}(X_1, \dots, X_7) &= \\
 &(\text{OR_4_4}(X_1, X_2, X_3, X_4) \wedge \text{OR_3_2}(X_5, X_6, X_7)) \vee \\
 &(\text{OR_4_3}(X_1, X_2, X_3, X_4) \wedge \text{OR_3_3}(X_5, X_6, X_7)).
 \end{aligned}$$

One needs a method to choose between the implementations in figures 12 and 13. Here is a pneumonic rule for making a choice. If one or two inputs arrive essentially later then one should use the implementation on figure 12 based on partition $7=5+2$. Otherwise, the implementation on figure 13 based on partition $7=4+3$ is probably optimal.

Parallel counters with 6, 5, and 4 inputs can be implemented according to the logic for the seven input parallel counter. Reducing the number of inputs decreases the area significantly and increases the speed slightly. It is advantageous to implement a six input parallel counter using partitions of 6, $3+3$ or $4+2$.

A second aspect of the present invention comprises a technique for multiplication and this will be described hereinafter.

Multiplication is a fundamental operation in digital circuits. Given two n -digit binary numbers

$$A_{n-1}2^{n-1} + A_{n-2}2^{n-2} + \dots + A_12 + A_0 \text{ and } B_{n-1}2^{n-1} + B_{n-2}2^{n-2} + \dots + B_12 + B_0,$$

their product

$$P_{2n-1}2^{2n-1} + P_{2n-2}2^{2n-2} + \dots + P_12 + P_0$$

has up to $2n$ digits. Logical circuits generating all P_i as outputs generally follow the scheme in figure 14. Wallace has invented the first fast architecture for a multiplier, now called the Wallace-tree multiplier (Wallace, C. S., *A Suggestion for a Fast Multiplier*, IEEE Trans. Electron. Comput. EC-13: 14-17 (1964)). Dadda has investigated bit behaviour in a multiplier (L. Dadda, *Some Schemes for Parallel Multipliers*, Alta Freq 34: 349-356 (1965)). He has constructed a variety of multipliers and most multipliers follow Dadda's scheme.

Dadda's multiplier uses the scheme in on figure 14. If inputs have 8 bits then 64 parallel AND gates generate an array shown in figure 15. The AND gate sign \wedge is omitted for clarity so that $A_i \wedge B_j$ becomes $A_i B_j$. The rest of figure 15 illustrates array reduction that involves full adders (FA) and half adders (HA). Bits from the same column are added by half adders or full adders. Some groups of bits fed into a full adder are in rectangles. Some groups of bits fed into a half adder are in ovals. The result of array reduction is just two binary numbers to be added at the last step. One adds these two numbers by one of fast addition schemes, for instance, conditional adder or carry-look-ahead adder.

This aspect of the present invention comprises two preferred steps: array deformation and array reduction using the parallel counter with the accordance with the first aspect of the present invention.

The process of array deformation will now be described.

Some parts of the multiplication array, formed by $A_i B_j$ such as on figure 15, have interesting properties. One can write simple formulas for the sum of the bits in these parts. Examples of such special parts are on figure 16. In general, chose an integer k , and those $A_i B_j$ in the array such that the absolute value of $i-j-k$ is less or equal to 1 comprise a special part.

Let S_i be the bits of the sum of all the bits of the form $A_i B_j$ shown on figure 1. Then

$$S_0 = A_0 \wedge B_0,$$

$$S_1 = (A_1 \wedge B_0) \oplus (A_0 \wedge B_1),$$

$$S_2 = (A_1 \wedge B_1) \oplus (A_1 \wedge B_1 \wedge A_0 \wedge B_0),$$

$$S_{2k+1} = (A_{k+1} \wedge B_k) \oplus (A_k \wedge B_{k+1}) \oplus (A_k \wedge B_k \wedge A_{k-1} \wedge B_{k-1})$$

for all $k > 0$,

$$S_{2k} = (A_k \wedge B_k) \oplus (A_{k-1} \wedge B_{k-1} \wedge$$

$$((A_{k+1} \wedge B_{k+1}) \vee (A_{k-1} \wedge B_{k-1} \wedge (A_{k+1} \vee B_{k+1}))))$$

for all $k > 1$.

These formulas show that the logic for summing the chosen entries in the array does not get large. Whereas if random numbers were summed the logic for the $(n + 1)^{\text{th}}$ bit is larger than the logic for the n^{th} bit.

Using these formulas, one can generate a different array. The shape of array changes. This is why it is called array deformation. These formulas are important because one can speed up a multiplication circuit by generating an array of a particular shape.

The array in figure 17 is for an 8-bit multiplication. The AND gate sign \wedge is omitted for clarity so that $A_i \wedge B_j$ becomes $A_i B_j$. Array deformation logic generates X, Y, and Z:

$$X = (A_1 \wedge B_6) \oplus (A_0 \wedge B_7),$$

$$Y = A_1 \wedge B_7 \wedge \neg(A_0 \wedge B_6),$$

$$Z = A_1 \wedge B_7 \wedge A_0 \wedge B_6.$$

The advantage of this array over one in figure 15 is that the maximal number of bits in a column is smaller. The array in figure 15 has a column with 8 bits. The array on figure 17 has 4 columns with 7 bits but none with 8 or more bits. The logic for the generation of X Y and Z is illustrated in figure 18. This logic can be used in parallel with the first two full adders (illustrated in Figure 2) in the array reduction step thus avoiding delays caused by additional logic.

Array reduction is illustrated in figure 17. The first step utilizes 1 half adder, 3 full adders, 1 parallel counter with 4 inputs, 2 parallel counters with 5 inputs, 1 parallel counter with 6 inputs, and 4 parallel counters with 7 inputs. The three parallel counters (in columns 7, 8, and 9) have an implementation based on $7=5+2$ partition. The bits X,

Y, and Z join the group of two in the partition. The counter in column 6 is implemented on $7=4+3$ partition. The counter in column 5 is based on $6=3+3$ partition. The remaining counters should not be partitioned. The locations of full adders are indicated by ovals. The half adder is shown by a rectangle.

An adder for adding the final two binary numbers is designed based on arrival time of bits in two numbers. This gives a slight advantage but it is based on common knowledge, that is conditional adder and ripple-carry adder.

Although in this embodiment the addition of two 8 bit numbers has been illustrated, the invention is applicable to any N bit binary number addition. For example for 16 bit addition, the array reduction will reduce the middle column height from 16 to 15 thus allowing two seven bit full adders to be used for the first layer to generate two 3 bit outputs and the left over input can be used with the other two 3 outputs as an input to a further seven input full adder thus allowing the addition of the 16 bits in only two layers.

The second aspect of the present invention can be used with the parallel counter of the first aspect of the present invention to provide a fast circuit.

The final counter of the first aspect of the present invention has other applications, other than used in the multiplier of the second aspect of the present invention. It can be used in RSA and reduced area multipliers. Sometimes, it is practical to build just a fragment of the multiplier. This can happen when the array is too large, for instance in RSA algorithms where multiplicands may have more than 1000 bits. This fragment of a multiplier is then used repeatedly to reduce the array. In current implementations, it consists of a collection of full adders. One can use 7 input parallel counters followed by full adders instead.

A parallel counter can also be used in circuits for error correction codes. One can use a parallel counter to produce Hamming distance. This distance is useful in digital

communication. In particular the Hamming distance has to be computed in certain types of decoders, for instance, the Viterbi decoder or majority-logic decoder.

Given two binary messages (A_1, A_2, \dots, A_n) and (B_1, B_2, \dots, B_n) , the Hamming distance between them is the number of indices i between 1 and n such that A_i and B_i are different. This distance can be computed by a parallel counter whose n inputs are

$$(A_1 \oplus B_1, A_2 \oplus B_2, \dots, A_n \oplus B_n).$$

The multiply-and-add operation is fundamental in digital electronics because it includes filtering. Given $2n$ binary numbers $X_1, X_2, \dots, X_n, Y_1, Y_2, \dots, Y_n$, the result of this operation is

$$X_1 Y_1 + X_2 Y_2 + \dots + X_n Y_n.$$

One can use the multiplier described to implement multiply-and-add in hardware.

Another strategy can be to use the scheme in figure 14. All partial products in products $X_i Y_i$ generate an array. Then one uses the parallel counter X to reduce the array.

In the present invention, one can use the parallel counter whenever there is a need to add an array of numbers. For instance, multiplying negative number in two-complement form, one generates a different array by either Booth recording (A. D. Booth, *A Signed Binary Multiplication Technique*, Q. J. Mech. Appl. Math. 4: 236-240 (1951)) or another method. To obtain a product one adds this array of numbers.

CLAIMS:

1. A parallel counter comprising:
a plurality of inputs for receiving a binary number as a plurality of binary inputs;
a plurality of outputs for outputting binary code indicating the number of binary ones in the plurality of binary inputs; and
a logic circuit connected between the plurality of inputs and the plurality of binary outputs and for generating each of the plurality of binary outputs as a symmetric function of the binary inputs.
2. A parallel counter according to claim 1 wherein said logic circuit is arranged to generate at least one of the binary outputs as a symmetric function of the binary inputs using exclusive OR logic for combining a plurality of sets of one or more binary inputs.
3. A parallel counter according to claim 2 wherein said logic circuit is arranged to logically AND members of each set of binary inputs and to logically exclusively OR the result of the AND operations.
4. A parallel counter according to claim 3 wherein said logic circuit is arranged to logically AND 2^i of the binary inputs in each set for the generation of the i^{th} binary output, where i is an integer from 1 to N , N is the number of binary outputs and i represents the significance of each binary output, each set being unique and the sets covering all possible combinations of binary inputs.
5. A parallel counter according to claim 3 wherein said logic circuit is arranged to logically AND members of each set of binary inputs, where each set is unique and the sets cover all possible combinations of binary inputs.
6. A parallel counter according to any preceding claim wherein said logic circuit is arranged to generate at least one of the binary outputs as a symmetric function of the

binary inputs using OR logic for combining a plurality of sets of one or more binary inputs.

7. A parallel counter according to claim 6 wherein said logic circuit is arranged to logically AND members of each set of binary inputs and to logically OR the result of the AND operations.

8. A parallel counter according to claim 7 wherein said logic circuit is arranged to logically AND 2^{N-1} of the binary inputs in each set for the generation of the N^{th} binary output, where N is the number of binary outputs and the N^{th} binary output is the most significant, each set being unique and the sets covering all possible combinations of binary inputs.

9. A parallel counter according to claim 7 wherein said logic circuit is arranged to logically AND members of each set of binary inputs, where each set is unique and the sets cover all possible combinations of binary inputs.

10. A parallel counter according to claim 1 wherein said logic circuit is arranged to generate a first binary output as a symmetric function of the binary inputs using exclusive OR logic for combining a plurality of sets of one or more binary inputs, and to generate an N^{th} binary output as a symmetric function of the binary inputs using OR logic for combining a plurality of sets of one or more binary inputs.

11. A parallel counter according to any preceding claim wherein said logic circuit is arranged to generate two possible binary outputs for a binary output less significant than the N^{th} binary output, as symmetric functions of the binary inputs using OR logic for combining a plurality of sets of one or more binary inputs where N is the number of binary outputs, the sets used for each possible binary output being of two different sizes which are a function of the binary output being generated; and said logic circuit including selector logic to select one of the possible binary outputs based on a more significant binary output value.

12. A parallel counter according to claim 11 wherein said logic circuit is arranged to generate the two possible binary outputs for the $(N-1)^{\text{th}}$ binary output less significant than the N^{th} binary output, as symmetric functions of the binary inputs using OR logic for combining a plurality of sets of one or more binary inputs, the sets used for each possible binary output being of size $2^{N-1} + 2^{N-2}$ and 2^{N-2} respectively and said selector logic being arranged to select one of the possible binary outputs based on the N^{th} binary output value.

13. A parallel counter according to any preceding claim wherein said logic circuit includes a plurality of subcircuit logic modules each generating intermediate binary outputs as a symmetric function of some of the binary inputs, and logic for logically combining the intermediate binary outputs to generate said binary outputs.

14. A parallel counter according to claim 13 wherein said subcircuit logic modules are arranged to use OR logic for combining sets of said some of said binary inputs.

15. A parallel counter according to claim 14 wherein said logic circuit includes one or more logic modules each for generating a binary output as a symmetric function of the binary inputs using executive OR logic for combining a plurality of sets of one or more binary inputs.

16. A logic circuit for multiplying two N bit binary numbers, the logic circuit comprising:

array generation logic for performing the logical AND operation between each bit in one binary number and each bit in the other binary number to generate an array of logical AND combination comprising an array of binary values, and for further logically combining values to generate the array in which the maximal depth of the array is below N bits;

array reduction logic for reducing the depth of the array to two binary numbers;
and

addition logic for adding the binary values of the two binary numbers.

17. A logic circuit according to claim 16 wherein said array generation logic is arranged to perform the further logical combination of values for values formed by the logical AND combination of each bit A_i of one binary number and each bit B_j of the other binary number, where $i-j-k \leq 1$, k is a chosen integer, and i and j are integers from 1 to N .

18. A logic circuit according to claim 16 or claim 17 wherein said array generation logic is arranged to logically AND combine each bit A_i of the first binary number with each bit B_j of a second binary number to generate said array comprising a sequence of binary numbers represented by said logical AND combinations, A_i AND B_j and to carry out further logical combination by logically combining the combination A_1 AND B_{N-2} , A_1 AND B_{N-1} where N is the number of bits in the binary numbers.

19. A logic circuit according to claim 18 wherein said array generation logic is arranged to combine the combinations A_1 AND B_{N-2} and A_0 AND B_{N-1} , using exclusive OR logic to replace these combinations, and to combine A_1 AND B_{N-1} and A_0 AND B_{N-2} to replace the A_1 AND B_{N-1} combination.

20. A logic circuit according to any one of claims 16 to 19 wherein said array reduction logic includes at least one of: at least one full adder, at least one half adder, and at least one parallel counter.

21. A logic circuit according to claim 20 wherein said array reduction logic includes at least one parallel counter according to any one of claims 1 to 15.

ABSTRACT:A PARALLEL COUNTER AND A MULTIPLICATION LOGIC CIRCUIT

A parallel counter comprises logic for generating output bits as symmetrical functions of the input bits. The parallel counter can be used in a multiplication circuit. A multiplication circuit is also provided in which an array of combinations of each bit of a binary number with each other bit of another binary number is generated having a reduced form in order to reduce the steps required in array reduction.

PRIOR ART

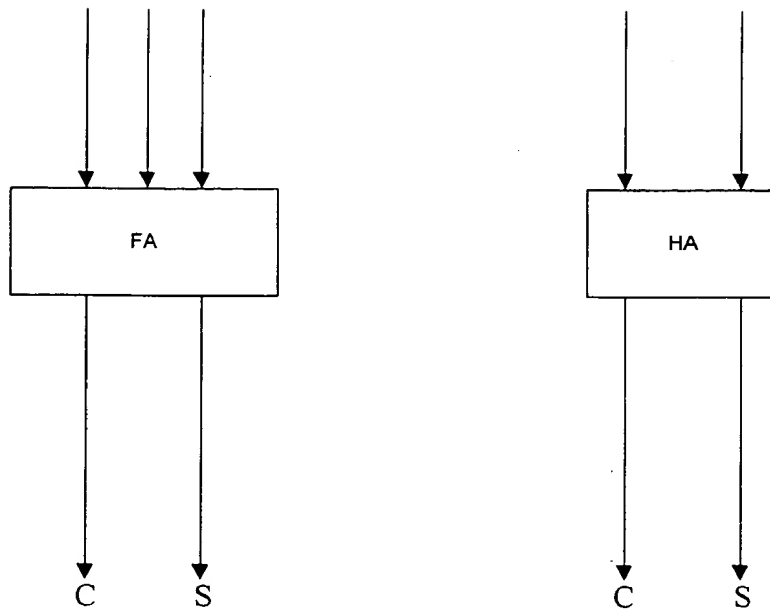


Fig 1

PRIOR ART

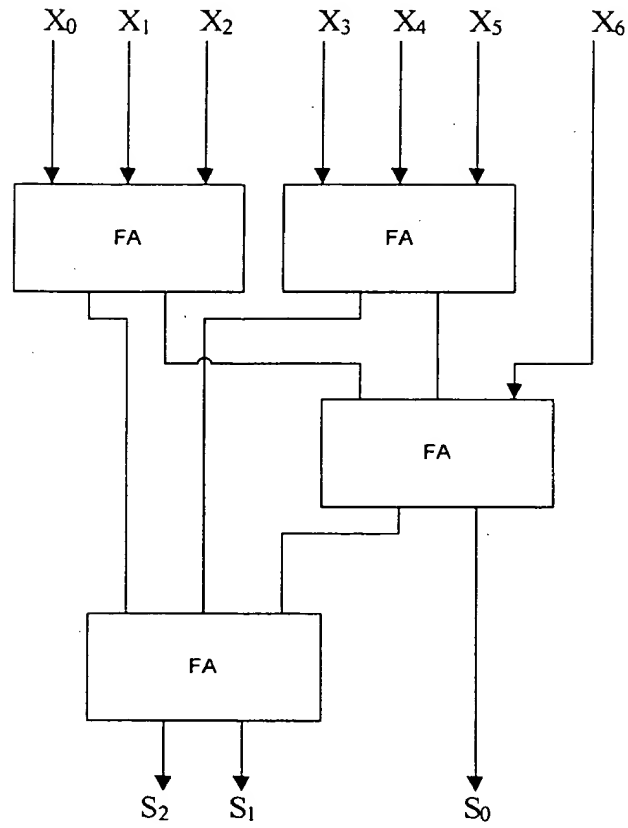


Fig 2

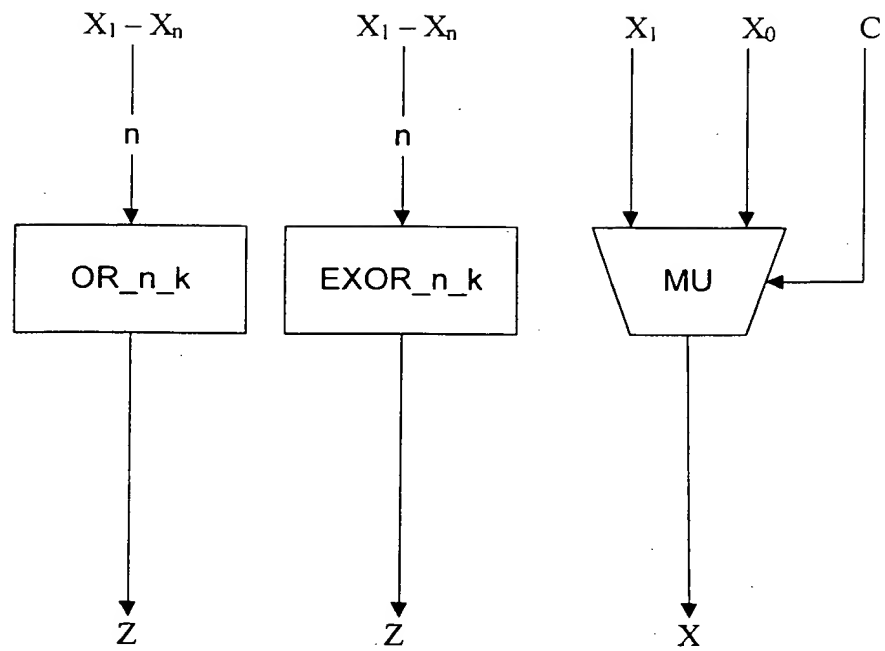


Fig 3

4/18

OR_3_1

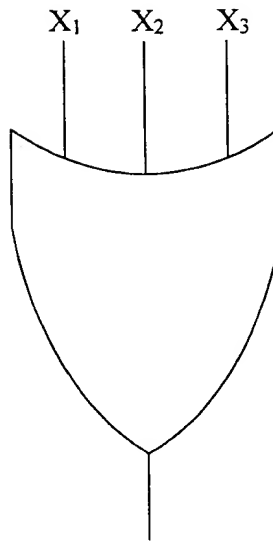


Fig 4

OR_4_1

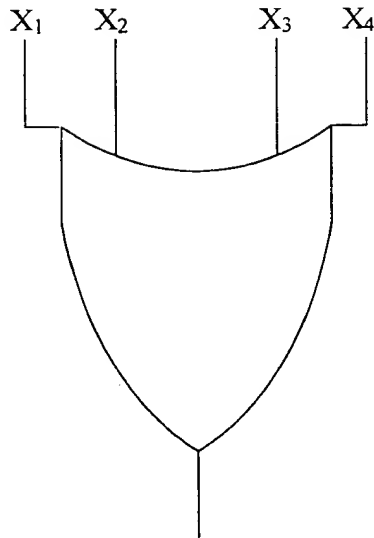


Fig 5

6/18

OR_5_1

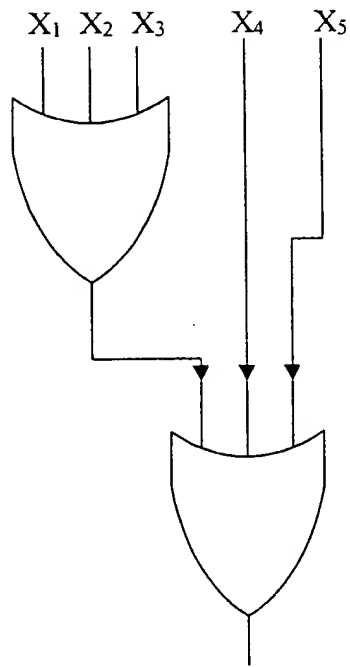


Fig 6

EXOR_7_1

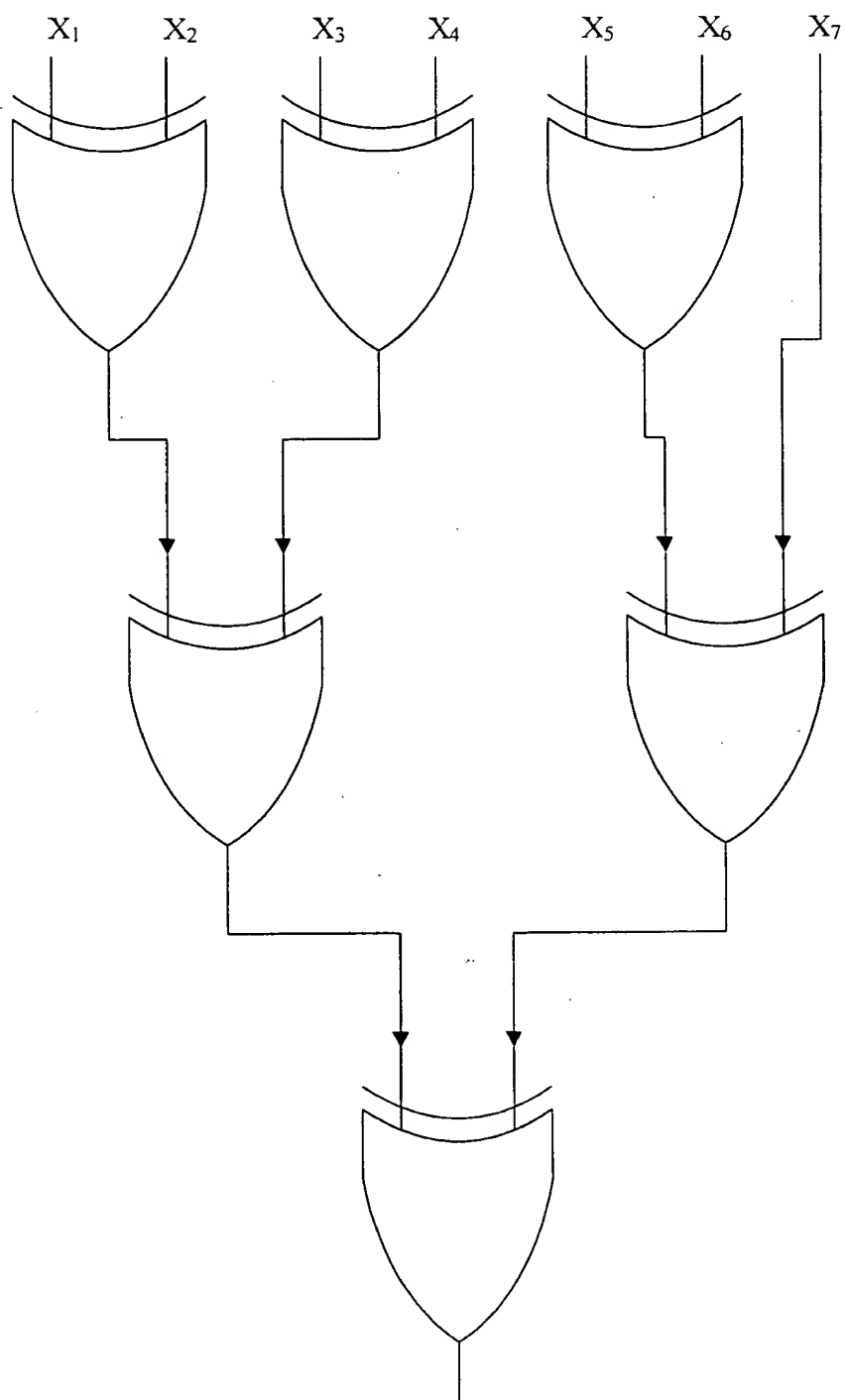


Fig 7

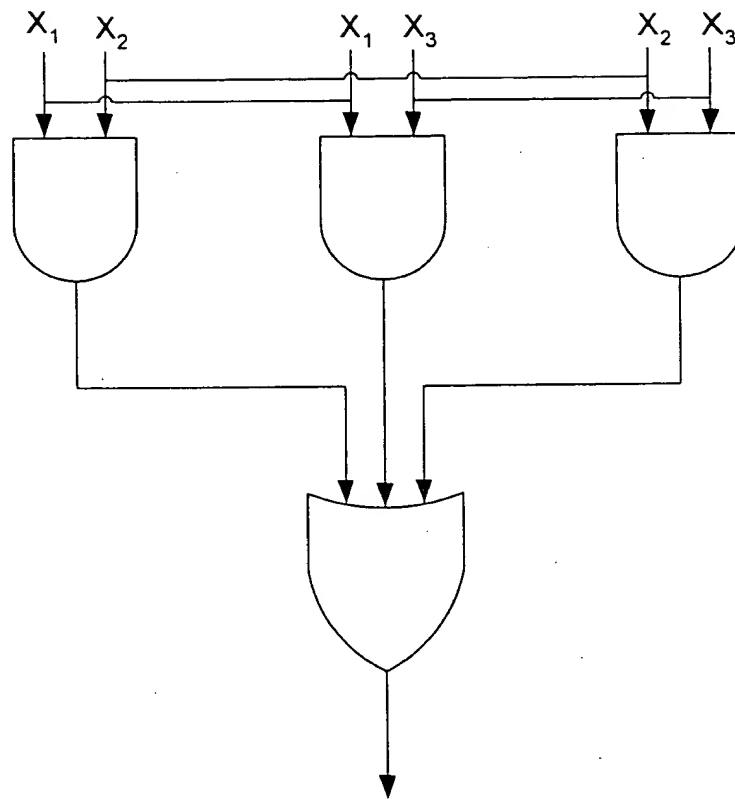


Fig 8

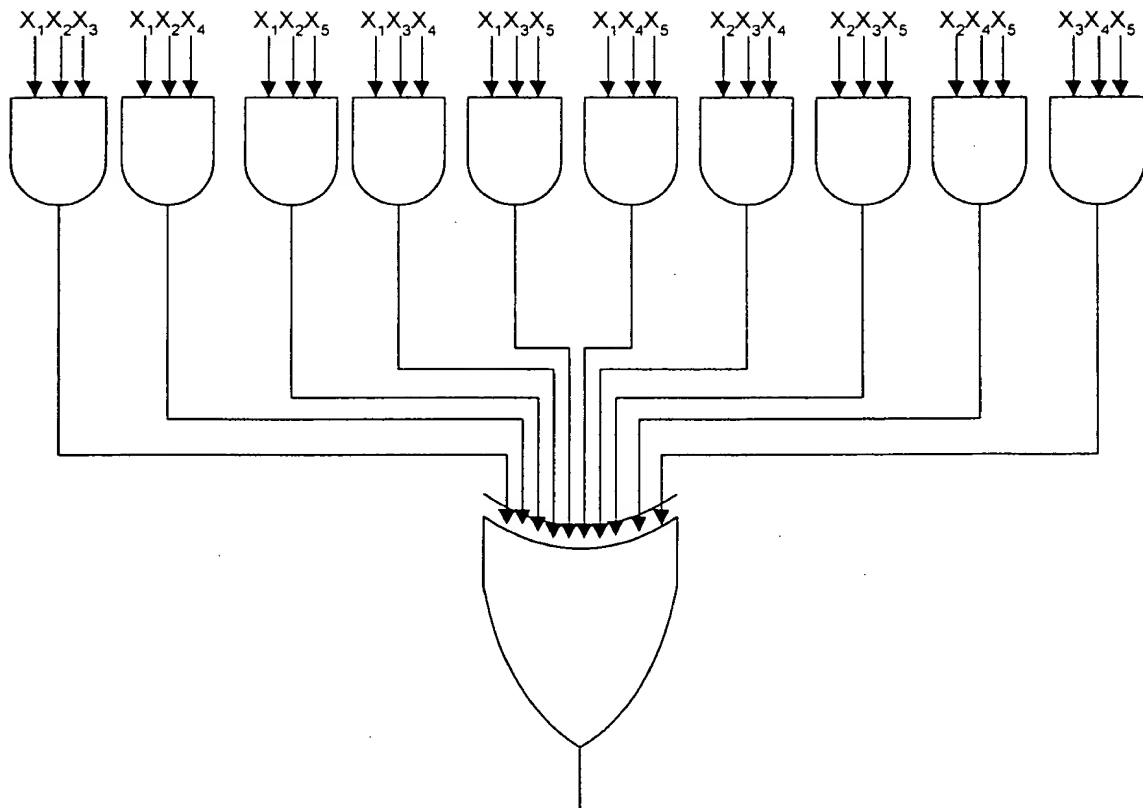


Fig 9

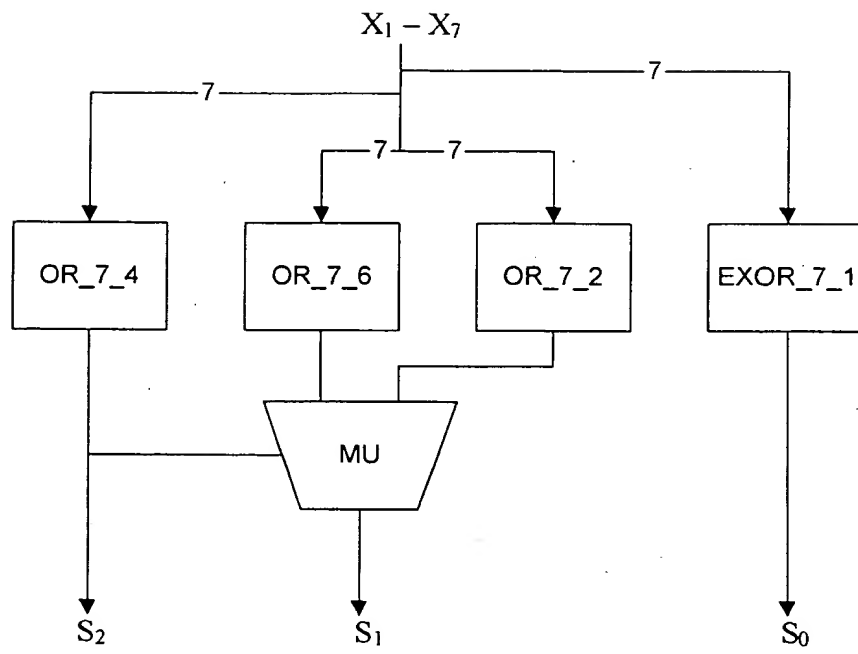


Fig 10

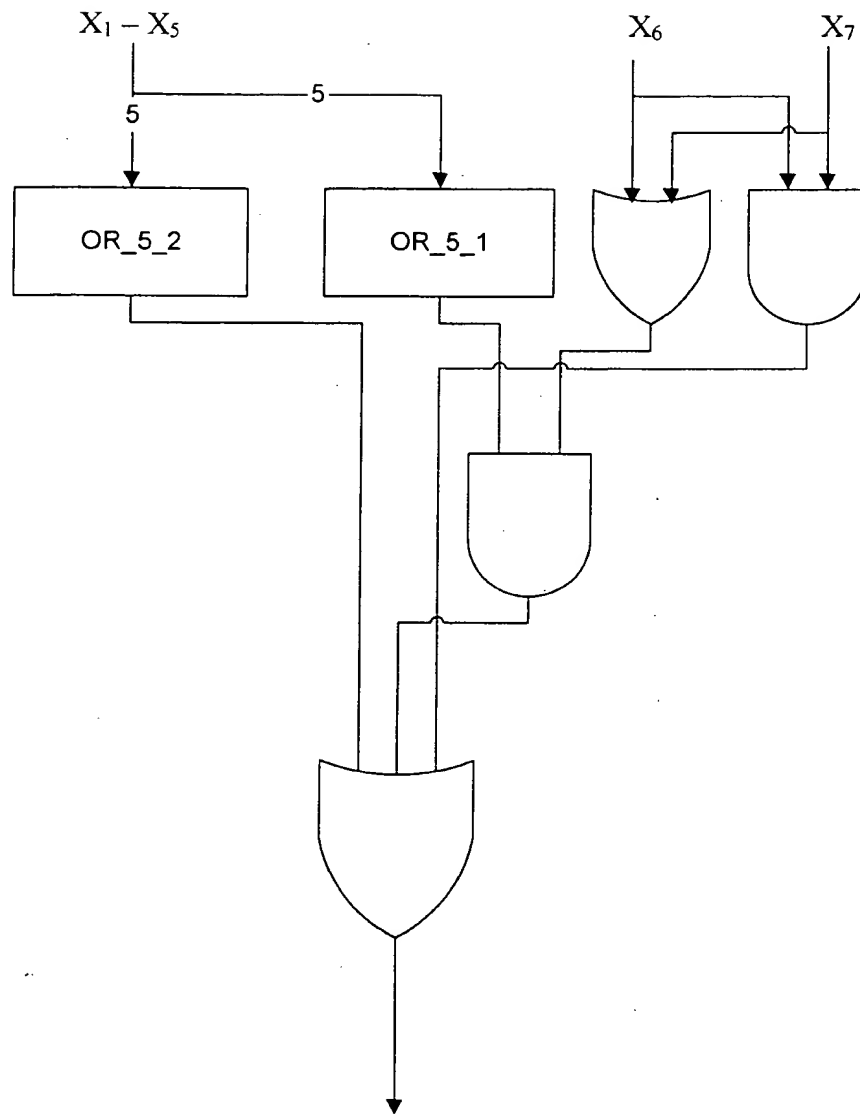


Fig 11

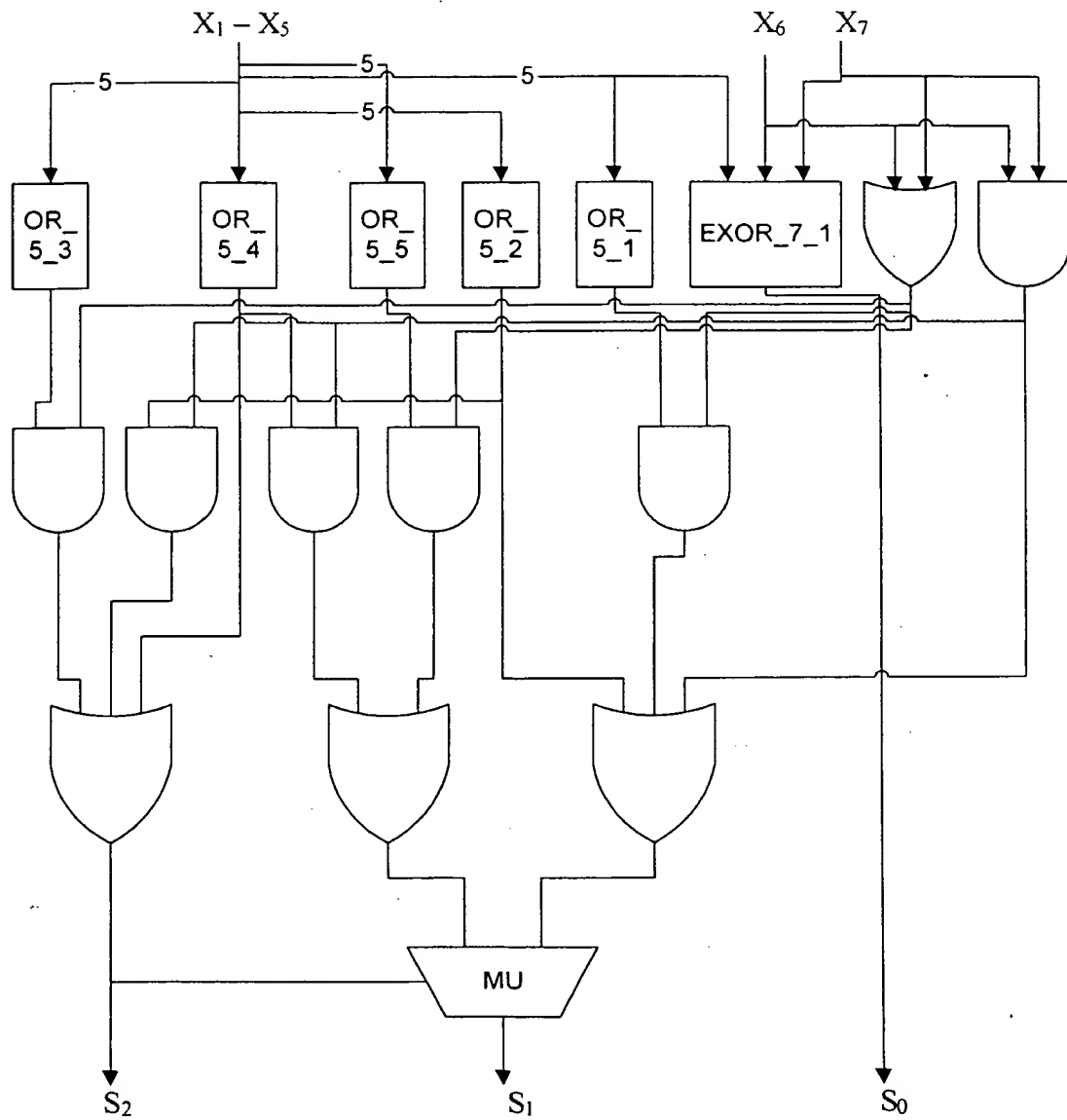


Fig 12

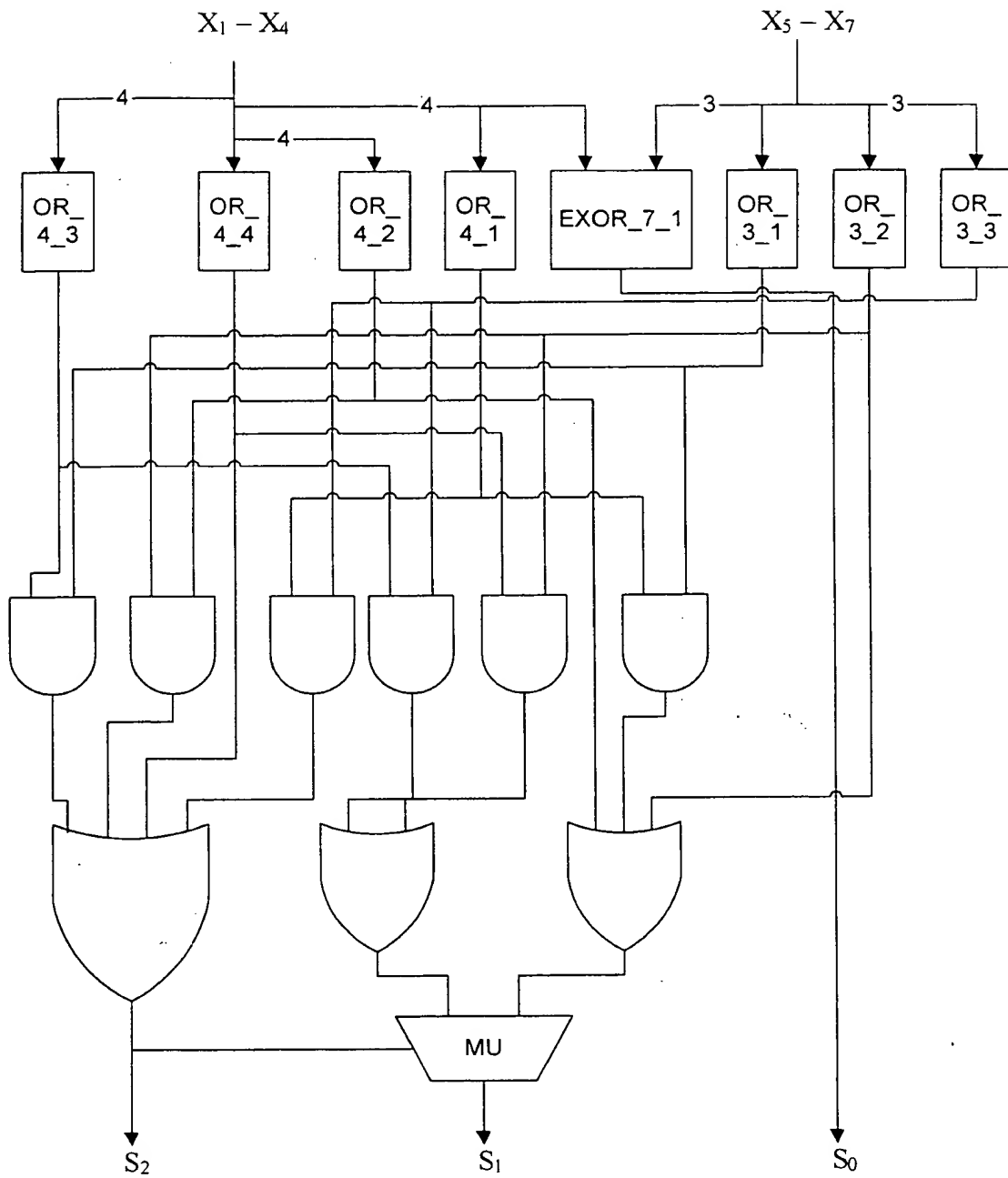


Fig 13

PRIOR ART

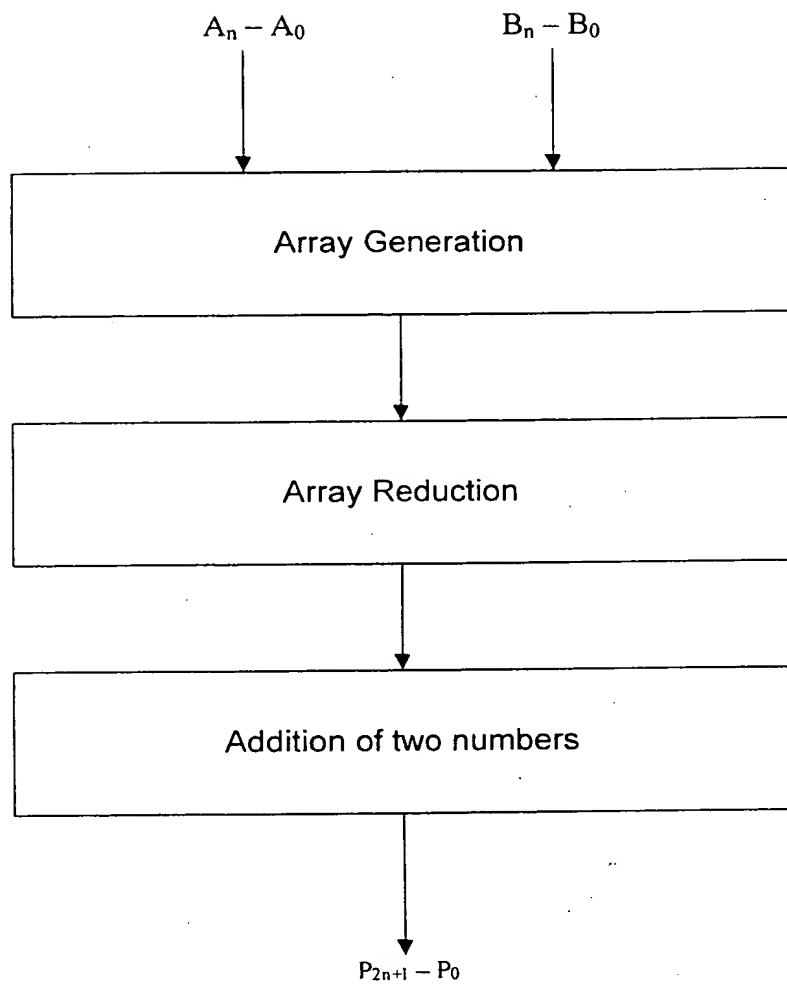


Fig 14

PRIOR ART

A_7B_0 A_6B_0 A_5B_0 A_4B_0 A_3B_0 A_2B_0 A_1B_0 A_0B_0
 A_7B_1 A_6B_1 A_5B_1 A_4B_1 A_3B_1 A_2B_1 A_1B_1 A_0B_1
 A_7B_2 A_6B_2 A_5B_2 A_4B_2 A_3B_2 A_2B_2 A_1B_2 A_0B_2
 A_7B_3 A_6B_3 A_5B_3 A_4B_3 A_3B_3 A_2B_3 A_1B_3 A_0B_3
 A_7B_4 A_6B_4 A_5B_4 A_4B_4 A_3B_4 A_2B_4 A_1B_4 A_0B_4
 A_7B_5 A_6B_5 A_5B_5 A_4B_5 A_3B_5 A_2B_5 A_1B_5 A_0B_5
 A_7B_6 A_6B_6 A_5B_6 A_4B_6 A_3B_6 A_2B_6 A_1B_6 A_0B_6
 A_7B_7 A_6B_7 A_5B_7 A_4B_7 A_3B_7 A_2B_7 A_1B_7 A_0B_7

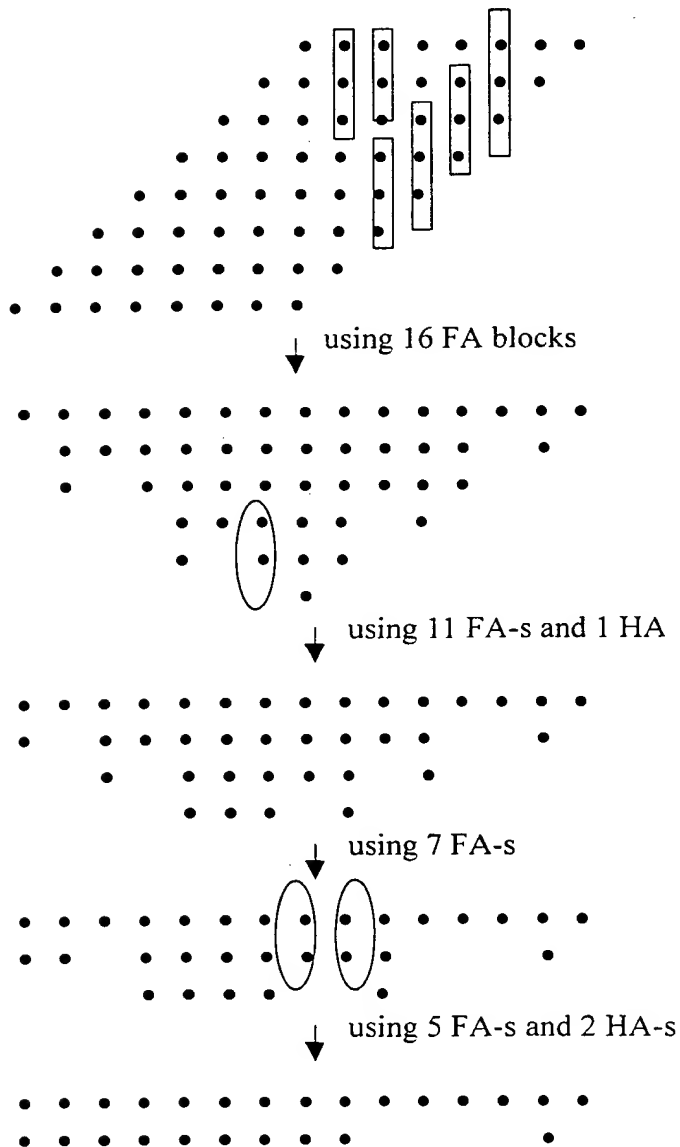


Fig 15

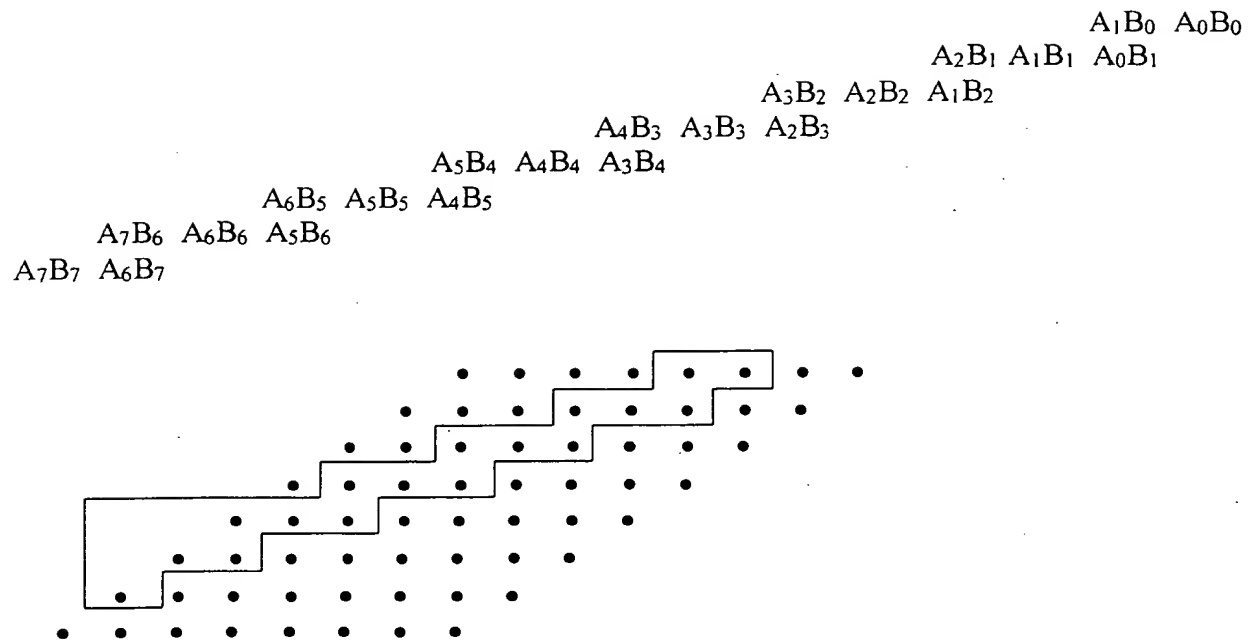


Fig 16

The diagram illustrates the reduction of a 16x16 dot grid to a 9x9 grid using hierarchical counters. The top grid is labeled with column numbers 15 down to 0. It features a 4x4 arrangement of 4x4 sub-grids. A 4x4 sub-grid on the left is highlighted with a rectangle and an oval, with an arrow pointing to the text "using 1 HA, 3 FA-s, and 8 higher parallel counters". A 4x4 sub-grid on the right is highlighted with an oval and an arrow pointing to the text "using 9 FA-s and 3 HA-s". The bottom grid is a 9x9 dot grid.

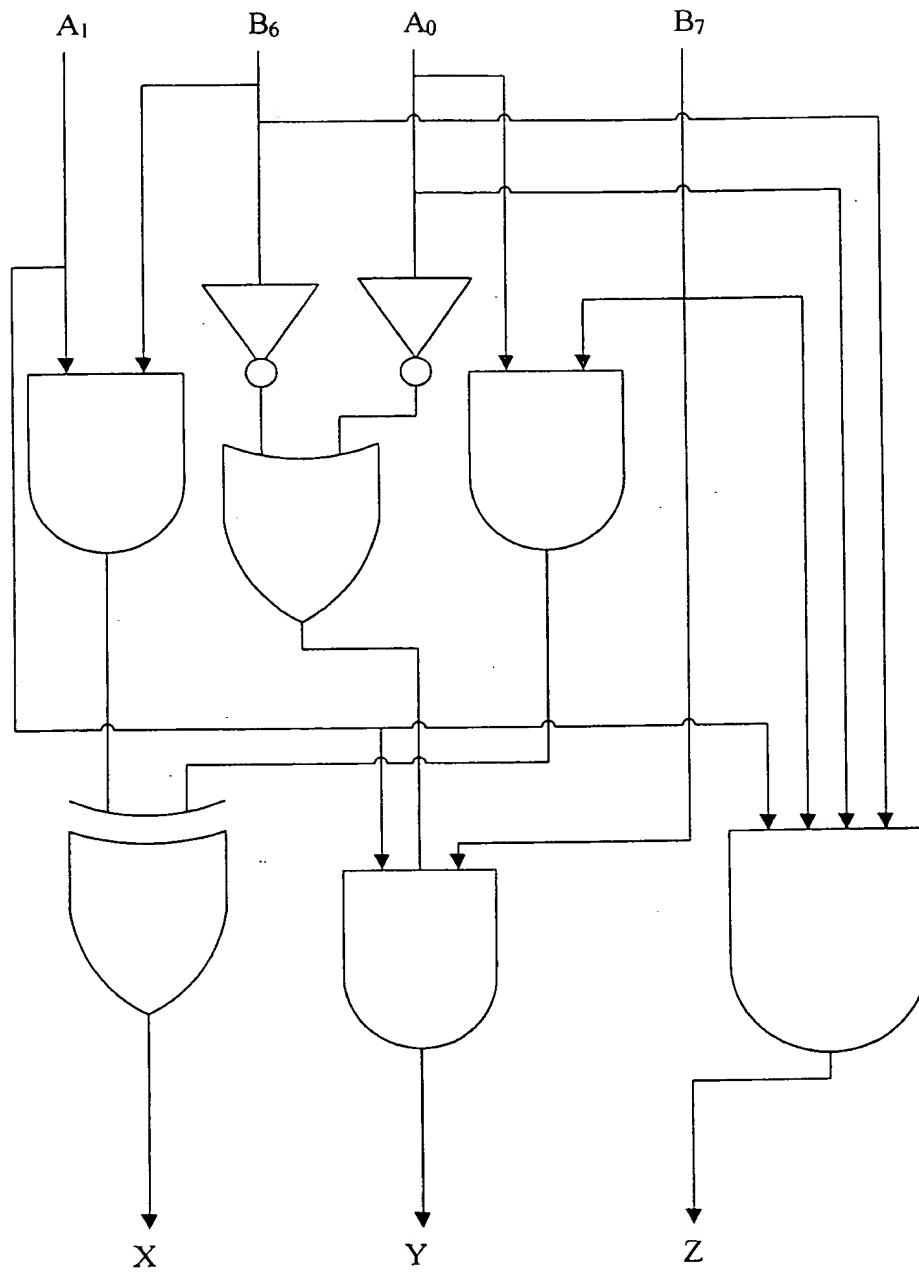


Fig 18

THIS PAGE BLANK (USPTO)